

CLOUDERA

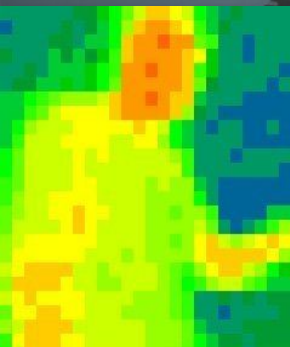
Utilizing Real-Time Transit Data for Travel Optimization

Tim Spann
Principal Developer Advocate

Sunday October 8, 2023
11:20 AM - 12:00 PM
Room 102

COMMUNITY
THE ASF CONFERENCE
CODE

Tim Spann



@PaasDev www.datainmotion.dev
github.com/tspannhw medium.com/@tspann
Principal Developer Advocate



Princeton Future of Data Meetup
ex-Pivotal, ex-Hortonworks, ex-StreamNative,
ex-PwC, ex-EY, ex-HPE.

Apache NiFi x Apache Kafka x Apache Flink

There are a lot of factors involved in determining how you can find our way around and avoid delays, bad weather, dangers and expenses. In this talk I will focus on public transport in the largest transit system in the United States, the MTA, which is focused around New York City. Utilizing public and semi-public data feeds, this can be extended to most city and metropolitan areas around the world. As a personal example, I live in New Jersey and this is an extremely useful use of open source and public data.

Once I am notified that I need to travel to Manhattan, I need to start my data streams flowing. Most of the data sources are REST feeds that are ingested by Apache NiFi to transform, convert, enrich and finalize it for usage in streaming tables with Flink SQL, but also keep that same contract with Kafka consumers, Iceberg tables and other users of this data. I do not need to many user interfaces to interopt with the system as I want my final decision sent in a Slack message to me and then I'll get moving. Along the way data will be visible in NiFi lineage, Kafka topic views, Flink SQL output, REST output and Iceberg tables.

Apache NiFi, Apache Kafka, Apache OpenNLP, Apache Tika, Apache Flink, Apache Avro, Apache Parquet, Apache Iceberg.

<https://github.com/tspannhw/FLaNK-MTA/tree/main>

<https://medium.com/@tspann/finding-the-best-way-around-7491c76ca4cb>

<https://medium.com/@tspann/open-source-streaming-talks-in-progress-3e75af8848b0>

<https://medium.com/@tspann/watching-airport-traffic-in-real-time-32c522a6e386>



Introduction

Overview

Sources & Code

Apache Kafka

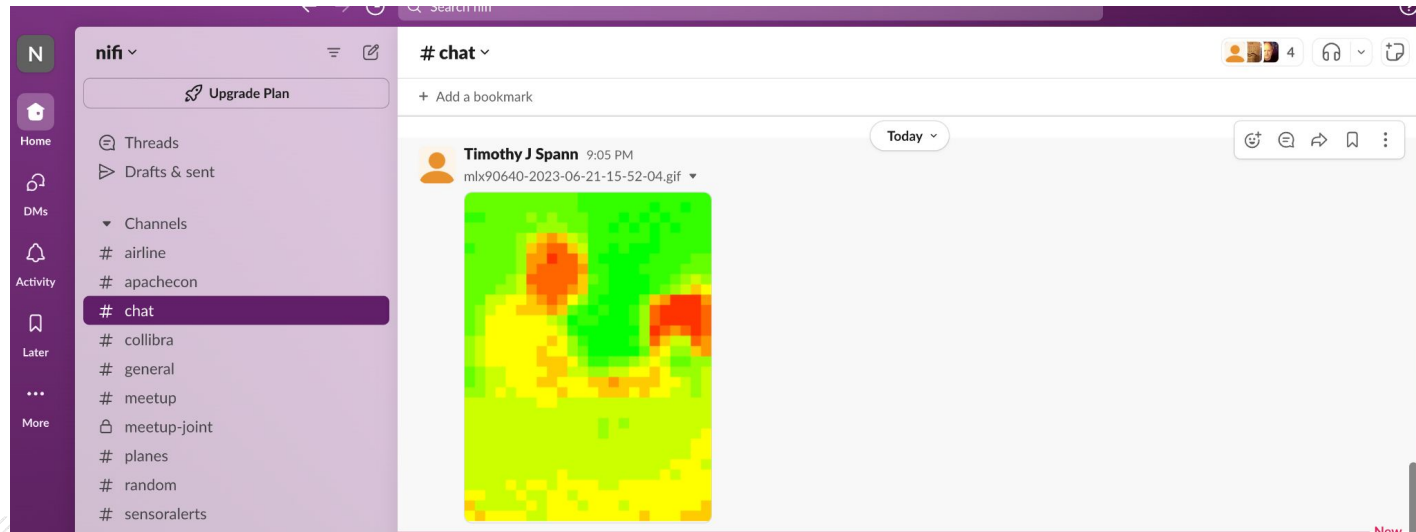
Apache Flink

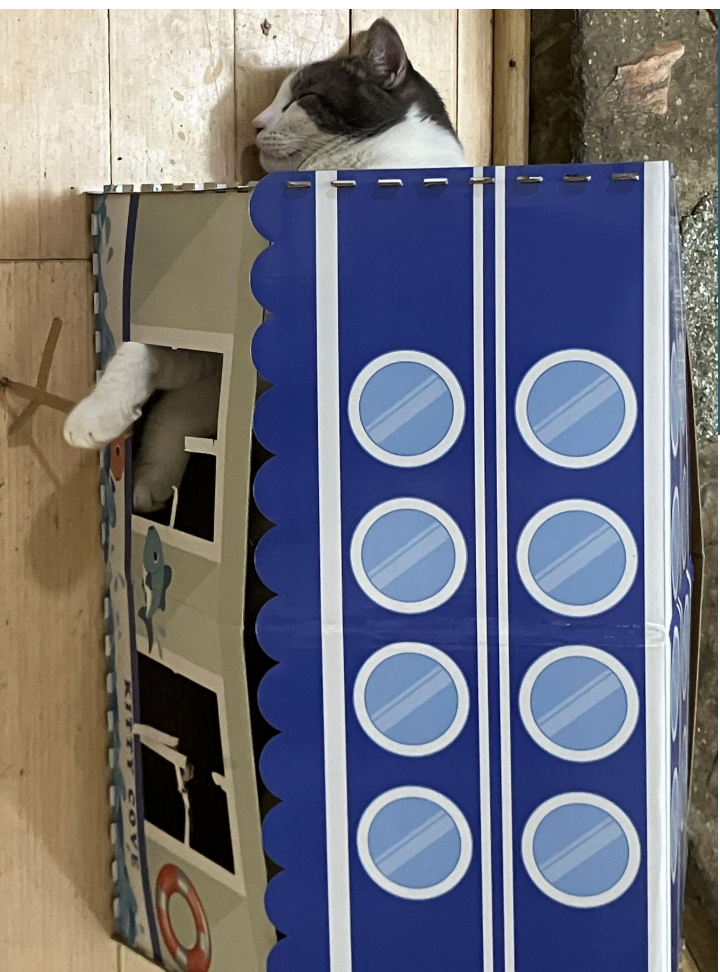
Apache NiFi

Demos



DATA OVERLOAD

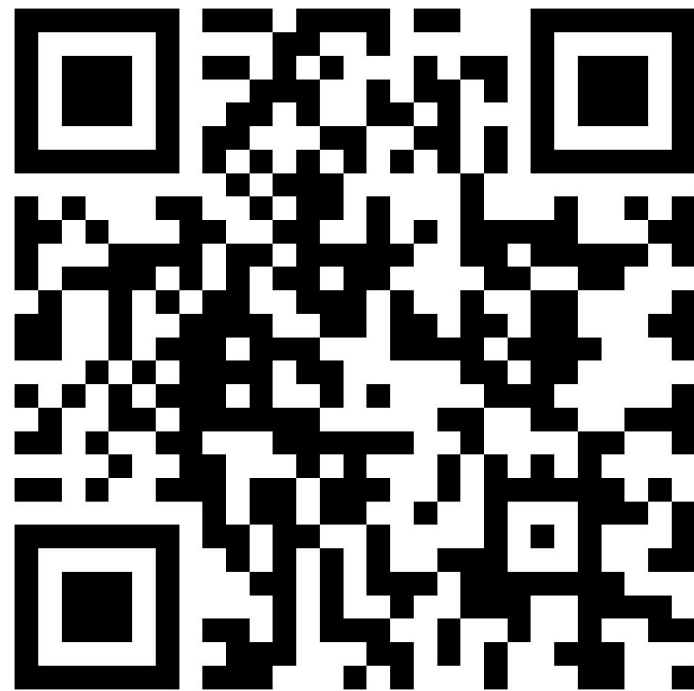




OUTGROWING ONE SERVER



USERS JUST WANTED
ONE TACO





OVERVIEW

Q. What is contain it everything?
A. Wisdom.

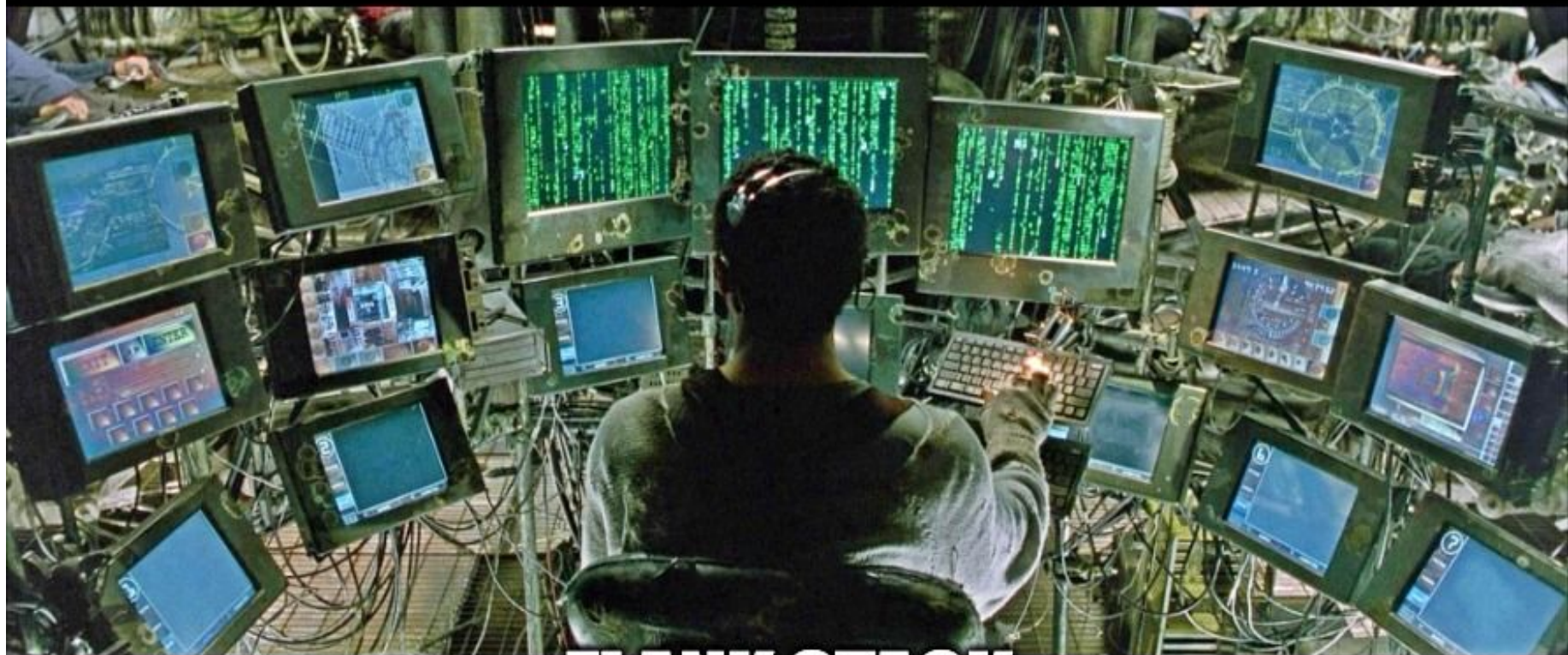
Trains, Planes and Automobiles +++

Information Needed	Data Feed(s)
Local weather conditions	<ul style="list-style-type: none">• XML, JSON, RSS
Mass transit status & alerts	<ul style="list-style-type: none">• XML, JSON, RSS
Regional highways & tunnels	<ul style="list-style-type: none">• GeoRSS, XML, ProtoBuf, JSON
Local social media	<ul style="list-style-type: none">• JSON
ADS-B Plane Data	<ul style="list-style-type: none">• JSON
Local air quality	<ul style="list-style-type: none">• JSON

REAL-TIME REQUIRES A PLATFORM



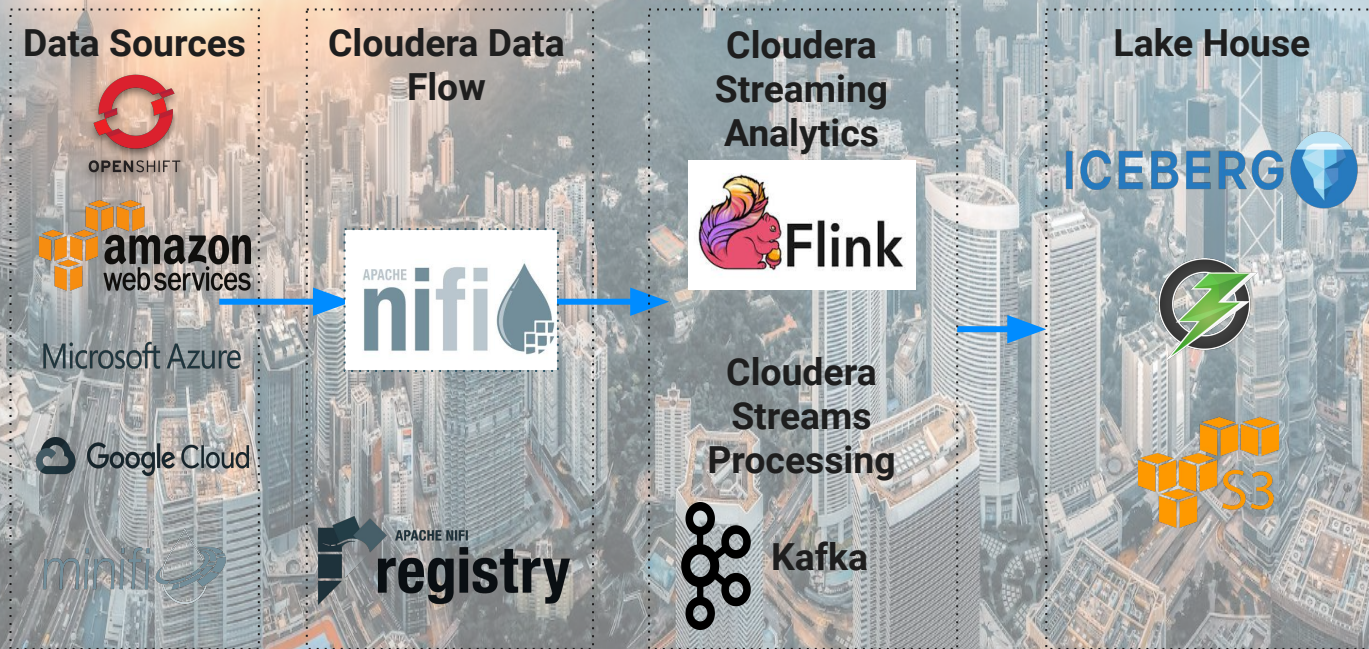
ALL THE TRANSIT DATA



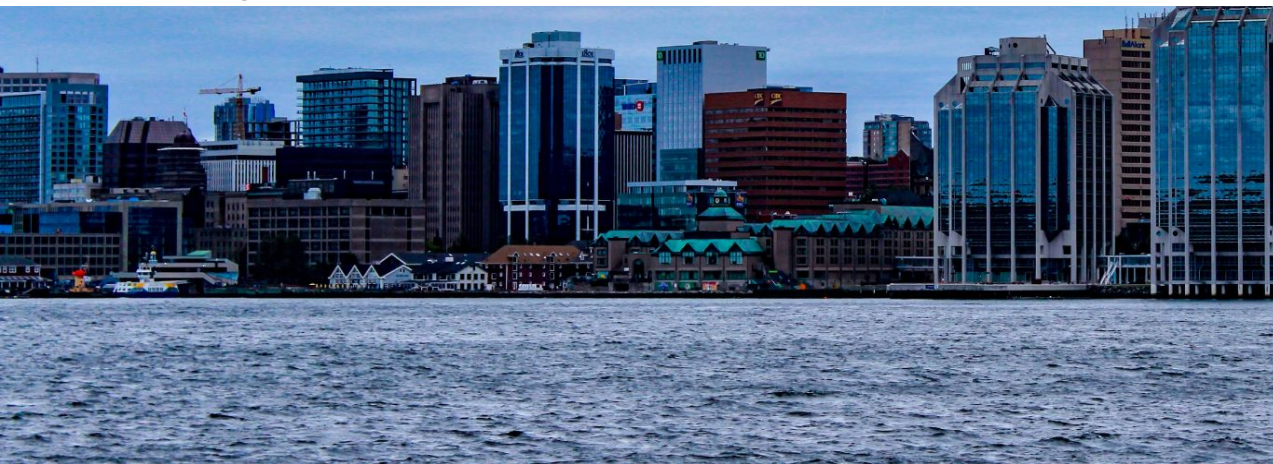
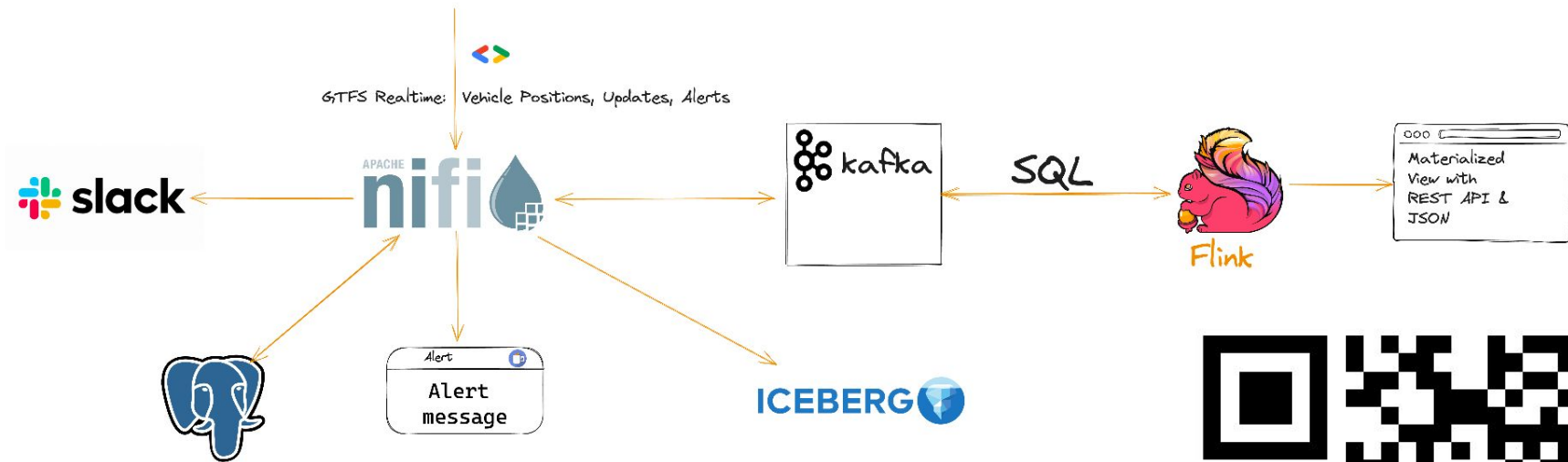
FLANK STACK

INGEST OF ALL TRANSIT DATA

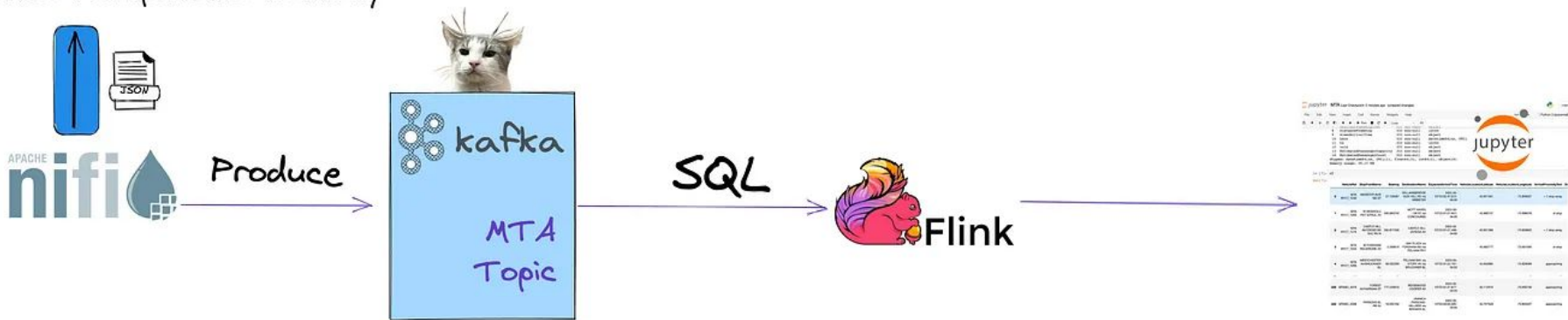
Run collection and streaming on any cloud, server, container or VM



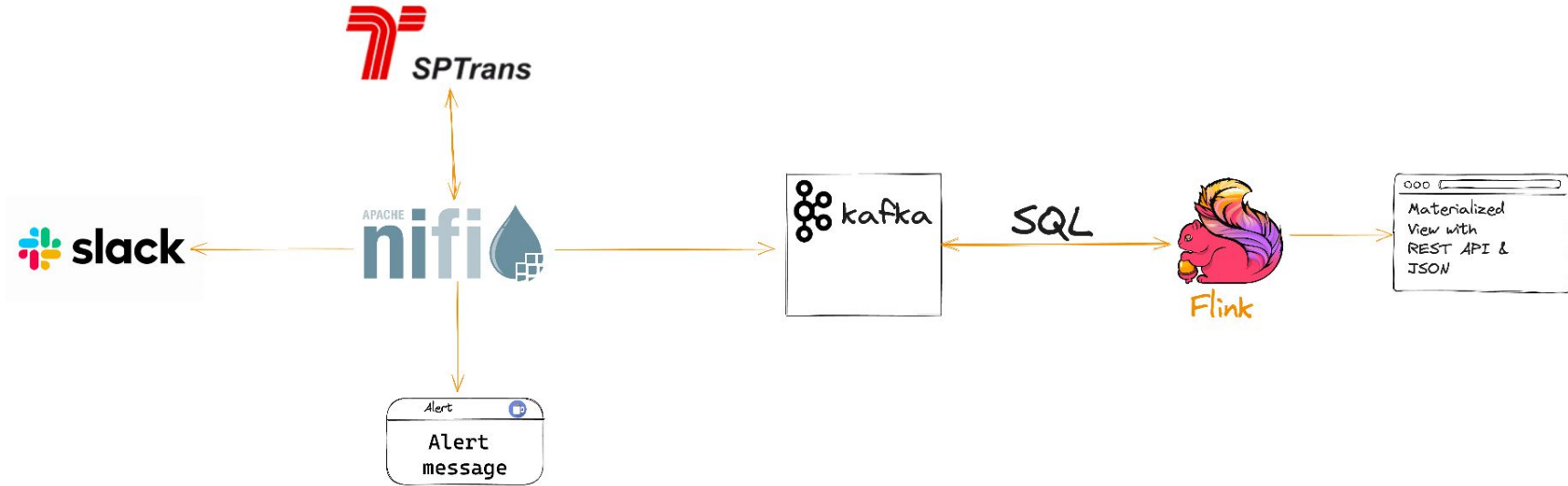
HALIFAX TRANSIT



Metropolitan Transportation Authority



CLOUDERA



Metropolitan Transportation Authority

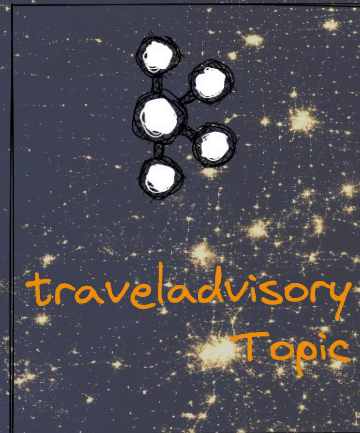


SQL





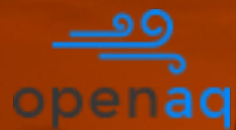
Produce



SQL



CLOUDERA



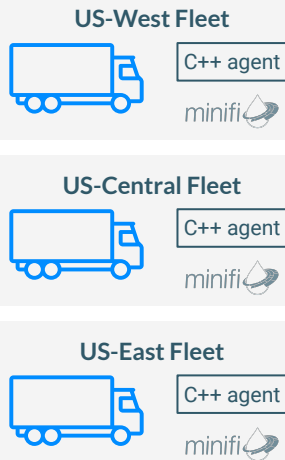
APACHE KAFKA



Apache Kafka

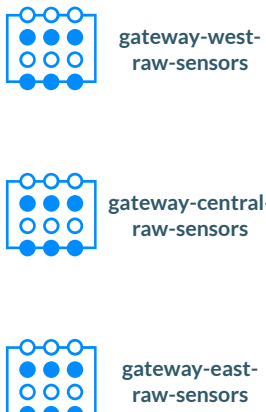
MINIFI

DATA COLLECTION AT THE EDGE



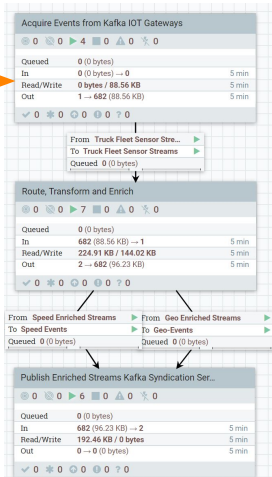
Apache Kafka

INGEST GATEWAY POWERED BY KAFKA



Apache NiFi

DATA FLOW APPS POWERED BY NIFI



Apache Kafka

KAFKA CLUSTER GEO 1 DATA SYNDICATE SERVICES



KAFKA CLUSTER GEO 2 DATA SYNDICATE SERVICES



Apache Flink

STREAMING ANALYTICS APPS

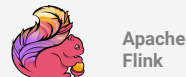
Micro Batch Analytics
Stream Analytics App



Micro Services
Stream Analytics App



Complex Low Latent
Stream Analytics App



APACHE FLINK



Apache Flink SQL

Democratize access to real-time data with just SQL



```
CREATE TABLE `ssb`.`Meetups`.`hfbloom` (  
  `generated_text` VARCHAR(2147483647),  
  `ts` VARCHAR(2147483647),  
  `x_compute_type` VARCHAR(2147483647),  
  `inputs` VARCHAR(2147483647),  
  `x_compute_time` VARCHAR(2147483647),  
  `x_inference_time` VARCHAR(2147483647),  
  `uuid` VARCHAR(2147483647),  
  `x_time_per_token` VARCHAR(2147483647),  
  `x_compute_characters` VARCHAR(2147483647),  
  `eventTimeStamp` TIMESTAMP(3) WITH LOCAL TIME ZONE METADATA FROM 'timestamp',  
  WATERMARK FOR `eventTimeStamp` AS `eventTimeStamp` - INTERVAL '3' SECOND  
) WITH (  
  'scan.startup.mode' = 'group-offsets',  
  'properties.request.timeout.ms' = '120000',  
  'properties.auto.offset.reset' = 'earliest',  
  'format' = 'json',  
  'properties.bootstrap.servers' = 'kafka:9092',  
  'connector' = 'kafka',  
  'properties.transaction.timeout.ms' = '900000',  
  'topic' = 'hfbloom',  
  'properties.group.id' = 'llmBloomProps'  
)
```


 0

 53,639 / 153.08 MB

 0

 0

 230

 831

 546

 160

 0

 0

 0

 0

 0

 22:26:28 EDT



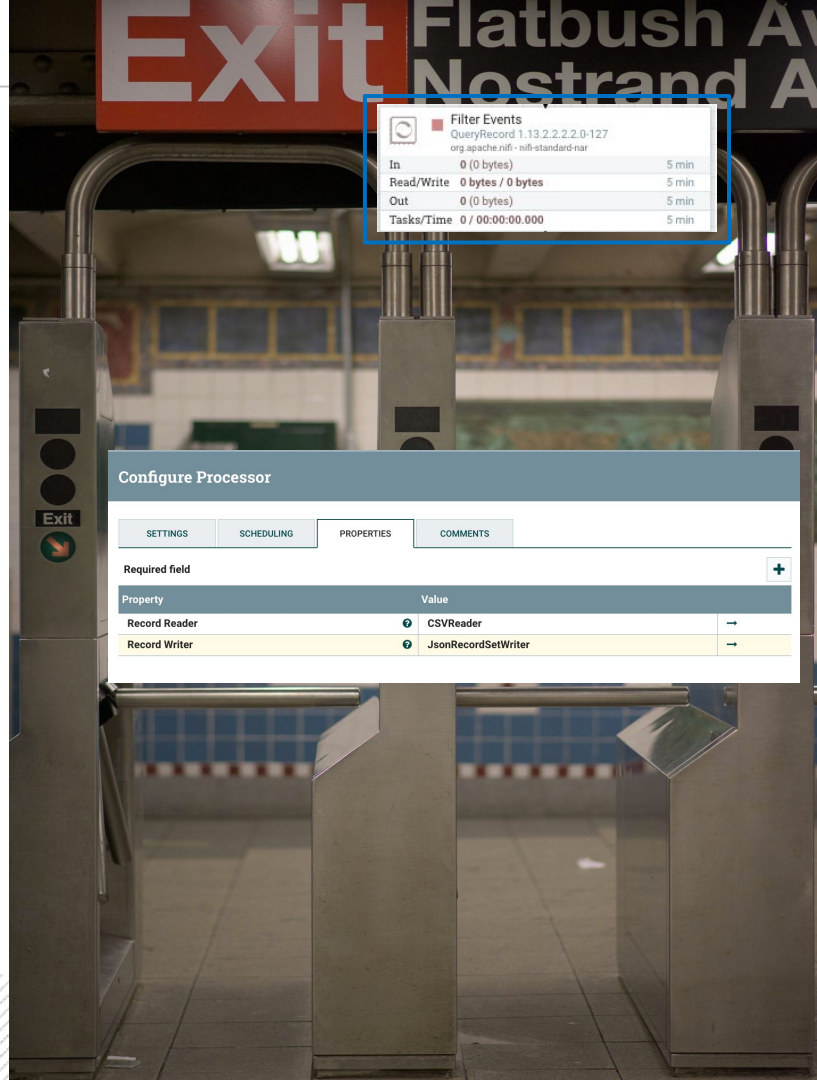


APACHE NIFI



RECORD-ORIENTED DATA WITH NIFI

- **Record Readers** - Avro, CSV, Grok, IPFIX, JSAN1, JSON, Parquet, Scripted, Syslog5424, Syslog, WindowsEvent, XML
- **Record Writers** - Avro, CSV, FreeFromText, Json, Parquet, Scripted, XML
- Record Reader and Writer support referencing a schema registry for retrieving schemas when necessary.
- Enable processors that accept any data format without having to worry about the parsing and serialization logic.
- Allows us to keep FlowFiles larger, each consisting of multiple records, which results in far better performance.



Filter Events		
QueryRecord 1.13.2.2.2.0-127		
org.apache.nifi - nifi-standard-nar		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Configure Processor

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field +

Property	Value
Record Reader	CSVReader
Record Writer	JsonRecordSetWriter

PROVENANCE

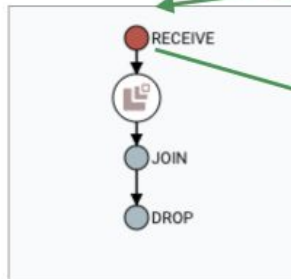
Displaying 13 of 104

Oldest event available: 11/15/2016 13:34:50 EST

Showing the most recent events.

Date/Time ▾	Type	FlowFile UUID	Size	Component Name	Component Type	
11/15/2016 13:35:03.8...	RECEIVE	379fc4f6-60e0-4151-9743-28...	44 bytes	ConsumeKafka	ConsumeKafka	→
11/15/2016 13:35:02.7...	RECEIVE	78f8c38b-89fc-4d00-a8d8-51...	44 bytes	ConsumeKafka	ConsumeKafka	→
11/15/2016 13:35:01.6...	RECEIVE	2bcd5124-bb78-489f-ad8a-7...	44 bytes	ConsumeKafka	ConsumeKafka	→

- Tracks data at each point as it flows through the system
- Records, indexes, and makes events available for display
- Handles fan-in/fan-out, i.e. merging and splitting data
- View attributes and content at given points in time



Provenance Event

DETAILS

ATTRIBUTES

CONTENT

Attribute Values

filename
328717796819631
No value previously set

kafka.offset
44815
No value previously set

kafka.partition
6
No value previously set

kafka.topic
nifi-testing
No value previously set

path
/
No value previously set

uuid
29571123-2852-4c44-8086-10542073-05

APACHE NIFI WITH PYTHON CUSTOM PROCESSORS

Python as a First Class Citizen

```
import cv2
import numpy as np
import json
from nifiapi.properties import PropertyDescriptor
from nifiapi.properties import ResourceDefinition
from nifiapi.flowfiletransform import FlowFileTransformResult

SCALE_FACTOR = 0.00392
NMS_THRESHOLD = 0.4 # non-maximum suppression threshold
CONFIDENCE_THRESHOLD = 0.5

class DetectObjectInImage:
    class Java:
        implements = ['org.apache.nifi.python.processor.FlowFileTransform']
    class ProcessorDetails:
        version = '0.0.1-SNAPSHOT'
        dependencies = ['numpy >= 1.23.6', 'opencv-python >= 4.6']

    def __init__(self, jvm=None, **kwargs):
        self.jvm = jvm

    # Build Property Descriptors
    self.model_file = PropertyDescriptor(
        name = 'Model File',
        description = 'The binary file containing the trained Deep Neural Network weights. Supports Caffe (*.caffemodel), TensorFlow (*.pb), Torch (*.t7, *.net), Darknet (*.weights), ' +
        'DLDLT (*.bin), and ONNX (*.onnx)',
        required = True,
        resource_definition = ResourceDefinition(allow_file = True)
    )

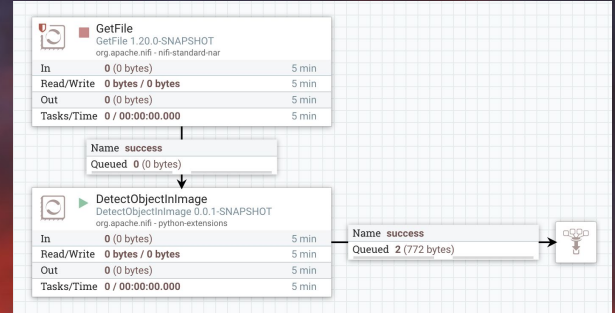
    self.config_file = PropertyDescriptor(
        name = 'Network Config File',
        description = 'The text file containing the Network configuration. Supports Caffe (*.prototxt), TensorFlow (*.pbtxt), Darknet (*.cfg), and DLDLT (*.xml)',
        required = False,
        resource_definition = ResourceDefinition(allow_file = True)
    )

    self.class_name_file = PropertyDescriptor(
        name = 'Class Names File',
        description = 'A text file containing the names of the classes that may be detected by the model. Expected format is one class name per line, new-line terminated.',
        required = True,
        resource_definition = ResourceDefinition(allow_file = True)
    )

    self.descriptors = [self.model_file, self.config_file, self.class_name_file]

    def getPropertyDescriptors(self):
        return self.descriptors

    def onScheduled(self, context):
        # read class names from text file
        class_name_file = context.getProperty(self.class_name_file.name).getValue()
        if class_name_file is None:
```



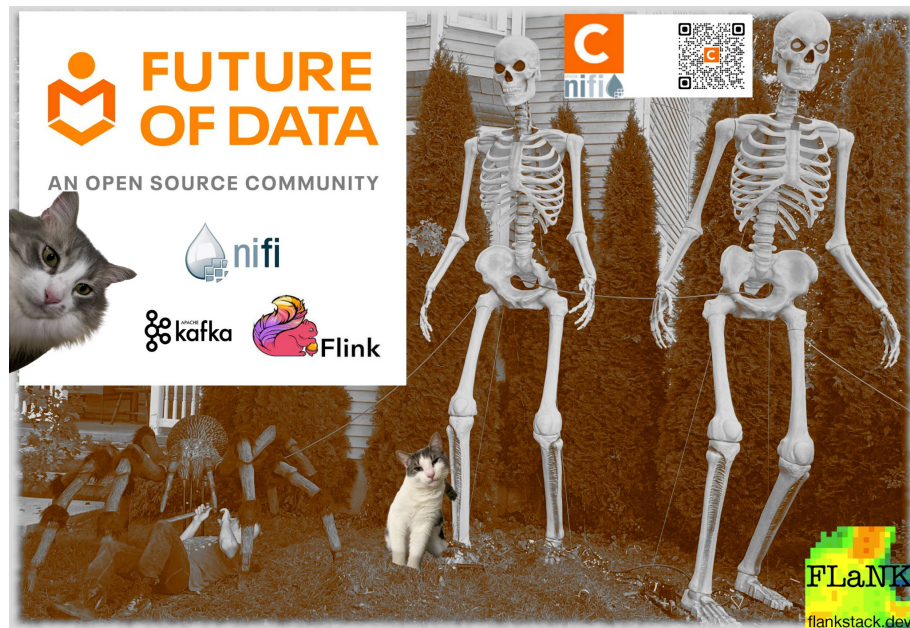
<https://github.com/apache/nifi/blob/614947e4ac6798ad80817e82514c39349d5faacb/nifi-docs/src/main/asciidoc/python-developer-guide.adoc>

Future of Data - NYC / Princeton + Virtual



<https://www.meetup.com/futureofdata-princeton/>
<https://www.meetup.com/futureofdata-newyork/>

From Big Data to AI to Streaming to Containers to Cloud to Analytics to Cloud Storage to Fast Data to Machine Learning to Microservices to ...



@PaasDev

FLaNK Stack Weekly



<https://bit.ly/32dAJft>



This week in Apache NiFi, Apache Flink, Apache Kafka, Apache Spark, Apache Iceberg, Python, Java, AI, ML, LLM and Open Source friends.

DEMO



TH[💧]N[🔗]Y[🦊]U

